Algorithmica (1986) 1: 93–109

Algorithmica

Edge-Skeletons in Arrangements with Applications

H. Edelsbrunner[1,2]

Abstract. An edge-skeleton in an arrangement $A(H)$ of a finite set of planes in E^3 is a connected collection of edges in $A(H)$. We give a method that constructs a skeleton in $O(\sqrt{n}\log n)$ time per edge. This method implies new and more efficient algorithms for a number of structures in computational geometry including order-k power diagrams in E^2 and space cutting trees in E^3.

We also give a novel method for handling special cases which has the potential to substantially decrease the amount of effort needed to implement geometric algorithms.

Key Words. Arrangements of planes, Power diagrams, Computational geometry, Asymptotic complexity, Dynamic data structures, Perturbation

1. Introduction. So-called (order-k) power diagrams in E^2 [A, IIM] and three-dimensional space cutting trees in E^3 [YDE, DE, EH] are structures designed for the algorithmic solution of several geometric problems (see Sections 5 and 6 for definitions of these notions). Once built, power diagrams can be used to find k nearest neighbour queries in the Laguerre metric, compute the area covered by at least k of some n given discs, etc. Space cutting trees in E^3 support a rather broad class of three-dimensional intersection queries [DE] including the following type:

> given a set of n points in E^3, determine how many or which are contained in a query tetrahedron.

The construction of both structures, and a few related ones, is non-trivial and only trivial and strikingly inefficient algorithms are known.

This paper demonstrates an algorithm that constructs the order-k power diagram for n circles in E^2 in $O(s_k(n)\sqrt{n}\log n)$ time and $O(n)$ extra storage, where $s_k(n)$ is the maximal number of regions that can occur in such a diagram. When all circles degenerate to points then the diagram is also known as the k-order Voronoi diagram. In this case, $s_k(n) = O(k(n-k))$, and the algorithm takes $O(kn\sqrt{n}\log n)$ time which improves the result of [L] who needs $O(k^2 n \log n)$ time and storage. (In fact, the storage is improved for all k, and the time is improved whenever $k = \omega(\sqrt{n})$.)

The trivial method for constructing a space cutting tree (as described in [YDE]) for n points in E^3 takes $O(n^7)$ time and $O(n)$ storage. We improve on this result

[1] Institutes for Information Processing, Technical University of Graz, Schiesstattgasse 4a, A-8010 Graz, Austria.

[2] Current address: Department of Computer Science, University of Illinois, 1304 West Springfield Avenue, Urbana, IL 61801, USA.

Received January 12, 1985; revised October 25, 1985, communicated by B. Chazelle.

with an algorithm that takes $O(t^2(n))$ time and $O(n)$ storage, where $t(n)$ is a poorly-understood function counting the number of edges in a combinatorial structure. The only bounds known for $t(n)$ are $O(n^3)$ and $\Omega(n \log n)$.

The results mentioned above are applications of a general method that computes so-called skeletons in arrangements of planes in E^3. We briefly sketch these concepts:

> a set H of n planes in E^3 dissects the space into a cell complex called the *arrangement* $A(H)$ *of* H.

$A(H)$ consists of various i-dimensional faces, for $0 \leq i \leq 3$, called cells ($i = 3$), facets ($i = 2$), edges ($i = 1$), and vertices ($i = 0$).

> a subset E of edges in $A(H)$ defines a *skeleton* if the union of their closures is connected.

The connection between skeletons and order-k power diagrams relies on a geometric transformation that maps circles in E^2 into planes in E^3. The diagram can then be obtained by vertical projection of a particular skeleton of the arrangement onto a horizontal plane. A similar connection exists between skeletons and space cutting trees: planes used to build such trees correspond by duality to points of particular skeletons in a dual arrangement.

Sections 2 and 3 describe an algorithm that computes a skeleton edge by edge with an effort of $O(\sqrt{n} \log n)$ time per edge. The algorithm is general in the sense that it does not depend on the particular skeleton; in principle, it can be used to construct any skeleton. Section 4 gives a new method for coping with degenerate cases like four planes intersecting in a common point, etc. We believe that the method can cut down the effort in implementing many known geometric algorithms.

How power diagrams correspond to skeletons in arrangements is explained in Section 5. Section 6 elaborates on the construction of two planes which cut two point-sets into respective balanced quaters; the method finds a skeleton in a dual arrangement. Finally, Section 7 offers a discussion of the contributions.

2. Skeletons and Eulerian Tours. This section gives formal definitions of arrangements and skeletons, and elaborates on the construction of eulerian tours in directed graphs which correspond to skeletons.

Let H be a finite set of non-vertical planes in E^3, that is, each plane in H intersects the x_3-axis in exactly one point. For a plane h: $x_3 = ax_1 + bx_2 + c$, we define

$$h^+ : x_3 > ax_1 + bx_2 + c, \quad \text{and}$$

$$h^- : x_3 < ax_1 + bx_2 + c.$$

A point p is said to be *above*, *on*, *below* h if h^+, h, h^- contains p, respectively. H dissects E^3 into a cell complex called the *arrangement* $A(H)$ *of* H. A *face* f is

a maximal set such that, for each h in H, f is contained in h^+, h, or h^-. If the affine hull[1] of f is k-dimensional[2], for $0 \leq k \leq 3$, then f is called a *cell*, *facet*, *edge*, *vertex of* $A(H)$, for $k = 3, 2, 1, 0$, respectively. Notice that all faces are relatively open and that no two faces intersect. Formulas for the maximal number of faces in an arrangement of n planes are classical results in the combinatorial literature [G, AW, etc.]:

PROPOSITION 2.1. Let H be a set of n planes in E^3. $A(H)$ contains at most $\binom{n}{3} + \binom{n}{2} + n + 1$ cells, $3\binom{n}{3} + 2\binom{n}{2} + n$ facets, $3\binom{n}{3} + \binom{n}{2}$ edges, and $\binom{n}{3}$ vertices.

The upper bounds of Proposition 2.1 are tight, and they are realized if $A(H)$ is *simple*, that is, if any three planes of H intersect in a common point and no four planes do so. Two faces are said to be *incident* if one is contained in the closure of the other, and the dimensionalities of their affine hulls differ by one. The maximum number of incidences is again realized if $A(H)$ is simple. In this case, a vertex, edge, facet is incident upon 6 edges, 4 facets, 2 cells, respectively.

Let E be a subset of edges in $A(H)$. $S(E) = \{x \in E^3 | x \in cle^3, e \in E\}$ is termed a *skeleton of* $A(H)$ if it is connected. The *edges* and *vertices* of $S(E)$ are the edges and vertices of $A(H)$ contained in $S(E)$. We adopt the convention that each unbounded edge of $S(E)$ is incident upon a vertex at infinity. $S(E)$ defines in a natural way a digraph $G(N, A)$ termed the *skeleton graph of* E or of $S(E)$:

> the set N of nodes stands in one-to-one correspondence with the set of vertices of $S(E)$. A directed arc $a = (v, w)$ is in A if E contains an edge incident upon the vertices (corresponding to) v and w.

We say that arc a *leads* from its *origin* v to its *destination* w, that v and w are *adjacent*, and that a is an *incoming* (*outgoing*) *arc of* w (v). The number of incoming (outgoing) arcs of some node v is called the *in-degree* $d^-(v)$ (*out-degree* $d^+(v)$) *of* v. By definition, arc $a = (v, w)$ is in A if and only if $-a = (w, v)$ is in A; consequently $d^-(v) = d^+(v)$, for every node v in N. A sequence $T = (a_0, \ldots a_k)$ of arcs in A is a *path* if the origin of a_i coincides with the destination of a_{i-1}, for $1 \leq i \leq k$, and T is a *tour* if furthermore the origin of a_0 equals the destination of a_k and $a_i \neq a_j$ if $i \neq j$. T is an *eulerian tour of* G if it is a tour and contains each arc of A. A necessary and sufficient condition for the existence of an eulerian tour is $\deg^-(v) = \deg^+(v)$, for each node v which is trivially true for any skeleton graph. We thus conclude:

OBSERVATION 2.2. There is an eulerian tour in every skeleton graph.

---

[1] The set of points x which can be described as $x = \Sigma a_i p_i$, with $p_i \in f$ and $\Sigma a_i = 1$ is the affine hull of f.

[2] The affine hull of f is k-dimensional if it can be described as the intersection of $3 - k$ but no fewer planes.

[3] *cl*e is an abbreviation for the closure of e.

On a high level of understanding, the algorithm of Section 3 which constructs a skeleton traverses an eulerian tour of the corresponding skeleton graph G. Since G is not completely known before the end of the traversal, the traversal of the eulerian tour must work with information local to a visited node and its incoming and outgoing arcs. One such algorithm is the traversal of a graph in depth-first search. A different method which avoids the use of recursion is described in the remainder of this section. To aid the presentation, we call an arc a *untouched* (w.r.t. the running algorithm) if neither a nor $-a$ have been traversed. Arc a is *touched* if $-a$ has been traversed but not a, and a is *forbidden* if a has been traversed.

Algorithm 1 (*Eulerian tour*). Initially, all arcs of A are untouched and v is an arbitrary node in N.

while not all outgoing arcs of v are forbidden *do if* there is an untouched arc $a = (v, w)$ *then* Traverse a, that is, set $v := w$. *else* Let $a = (v, w)$ be the non-forbidden arc with origin v such that a has been touched the latest, and traverse a, that is, set $v := w$. *endif endwhile.*

We argue about correctness:

LEMMA 2.3. Algorithm 1 traverses an eulerian tour of the skeleton graph $G = (N, A)$.

PROOF. Evidently, Algorithm 1 determines a path T in G. Also each arc is in T at most once, and T is a tour since $d^-(v) = d^+(v)$, for each node v, guarantees that T ends in the node where it starts. By the following argument, T is eulerian if no touched arc remains:

> Call an arc a in A *saturated* if a and $-a$ are forbidden. Suppose that no arc in A remains touched and that T fails to be eulerian. Then there is a node v with saturated and untouched incoming arcs which contradicts the control flow of Algorithm 1.

Let now a be the last arc in T which is not saturated, so $-a$ is not in T, and if arc b follows a in T then $-b$ is in T. Let v be the destination of a. By the control flow of Algorithm 1, an outgoing arc b of v can follow a in T only if b was untouched at the time of its traversal; so also $-b$ follows a in T. Since this is true for each outgoing arc of v following a in T, T ends in v; a contradiction since $-a$ is a non-forbidden outgoing arc of v. □

To turn Algorithm 1 into an algorithm that actually constructs a skeleton, we need a procedure that distinguishes edges of the skeleton from other edges in the arrangement (this procedure turns the general method into one that computes a particular skeleton), and a procedure that traverses arcs (note that arcs are given only implicitely by the planes).

3. Constructing Skeletons. This section details the construction of a skeleton without addressing the problem of recognizing edges of the desired skeleton. The description is necessarily incomplete as several components of the algorithm depend on the particular skeleton to be built; Sections 5 and 6 specify such components which lead to methods for computing order-k power diagrams in E^2 and dissecting planes in E^3. In this section, we assume that the arrangement considered is simple; the general case is treated in Section 4.

The algorithm makes use of two data structures that support the various actions taken:

1. The (currently known part of the) skeleton is stored in an adjacency structure ADJ of the corresponding skeleton graph. A dictionary is exploited to keep this representation augmentable.
2. The set of planes is organized in a data structure PEN which supports so-called penetration queries, insertions, and deletions.

We consider the first structure. Let $G = (N, A)$ be the skeleton graph which is not completely known before the computation ends. At any point in time, we let N' denote the set of nodes visited by the algorithm, and $A' = N' \times N$. Only nodes in N' and arcs in A' are stored in ADJ. Note that not every arc in A' is also traversed by the algorithm: An arc in $N' \times (N - N')$ is necessarily untouched, while an arc in $N' \times N'$ can be of any type. For this reason, untouched, touched, and forbidden arcs are carefully distinguished. We proceed with the detailed specification of ADJ (see Fig. 1):

(a) Each node v of N' is stored in a record which contains an array of six elements, each capable of storing one outgoing arc of v (every vertex in a simple arrangement is incident upon at most six edges of any skeleton).
(b) Each arc a of A' holds its status (untouched, touched, or forbidden), if it is touched, then it indicates the number of arcs with the same origin which became touched before a, and unless it is untouched it stores a pointer to its destination.
(c) Each node stores the three planes which define the corresponding vertex, and each arc stores the two planes which contain the corresponding edge.
(d) All records representing nodes in N' are organized in a dictionary that discriminates, say, by sets of containing planes.

Fig. 1(b) illustrates ADJ for the incomplete skeleton graph shown in (a): forbidden, touched, and untouched arcs are indicated by full, dotted, and broken lines, respectively. The adjacency lists with sets of containing planes omitted are displayed in Fig. 1(b). Clearly, the status of an arc can be changed in constant time. Furthermore, given a touched or forbidden arc a, arc $-a$ can be found in constant time since it is one of at most six outgoing arcs of the destination of a. By standard implementations of a dictionary, a node, specified by the three planes which intersect at the corresponding vertex, can be found or inserted into ADJ in logarithmic time (see [K] or [AHU]).

The second data structure PEN supports *penetration queries* defined as follows:

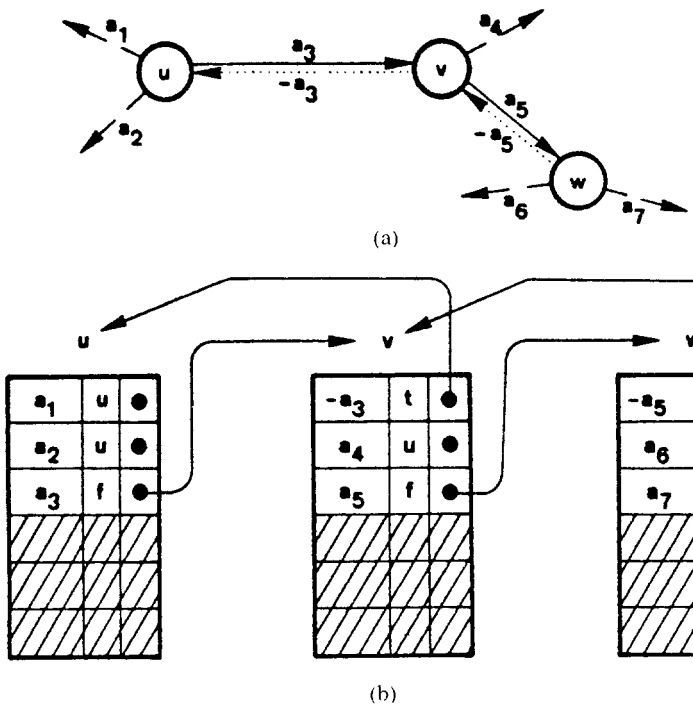

Fig. 1. Representing a skeleton graph. (a) Incomplete skeleton graph. (b) Adjacency lists.

Let H be a set of n planes that define n open halfspaces with non-empty common intersection T. A query is specified by a point p in T and a non-zero vector v, and asks for a plane that is hit first as p moves in the direction of v.

[EM] call this the penetration search problem and provide a solution that requires $O(n)$ storage, $O(\log n)$ time for a query, and $O(n \log n)$ time for its construction. Applying a general dynamization technique of [MO, vLW] yields

PROPOSITION 3.1. For a current number of n halfspaces, there is a data structure that takes $O(n)$ storage and $O(\sqrt{n} \log n)$ time for a penetration query and an insertion or deletion of a halfspace.

We remark only marginally that a technique of [GK] can be used to speed up queries, insertions and deletions to $O(\sqrt{n \log n})$ which costs $O(n \log \log n)$ storage, however.

A formal description of the algorithm is given below. Without confusion, little distinction is drawn between vertices (edges), corresponding nodes (arcs), and records which represent them in storage. On a high level of understanding, the algorithm traverses an eulerian tour of $G = (N, A)$.

The determination of sets of planes that contain vertices or edges deserves no mention in the outline; in all cases, it is either trivial or external to the algorithm.

Algorithm 2 (*constructing skeletons*). Initially, find some vertex v of $S(E)$ and all incident edges in E. Add the corresponding node v and all outgoing arcs of v (as untouched arcs) to ADJ. Construct PEN for all open halfspaces which are bounded by a plane in H and contain v.

while v has an untouched or touched arc *do*
  if no arc of v is untouched *then*
    Mark the latest touched arc (v, w) as forbidden.
  else {v has an untouched arc}
    Mark any untouched arc a of v as forbidden. Pose a penetration query for point v and the vector defined by a; this yields a plane which intersects the line that contains the edge corresponding to a in the destination w of a.
    if w is not yet in ADJ *then*
      Determine the outgoing arcs of w and add w and these arcs to ADJ.
    endif;
    Change the status of $-a$ to touched and record w as the destination of a.
  endif;
  Update PEN so that w is in the common intersection and v is not, that is, delete (insert) the unique open half-space that is bounded by a plane through w (v) and which contains v (w). Traverse (v, w), that is, set $v := w$.
endwhile.

The complexity of the algorithm depends heavily on the complexity of its various components. Let n be the cardinality of H and m the one of E. Then we define

$S_I(n)$ ($T_I(n)$) $\cdots$ the amount of storage (time) needed for finding an initial vertex,
$S_E(n)$ ($T_E(n)$) $\cdots$ the storage (time) required for determining all edges in E incident to some given vertex,
$S_P(n)$ $\cdots$ the storage taken by PEN, and
$T_P(n)$ $\cdots$ the maximal time needed for a query, insertion, or deletion in PEN.

The construction of one arc including its destination costs $O(\log m)$ time for manipulations in ADJ, $T_E(n)$ time for finding new outgoing arcs, and $T_P(n)$ time for a query, an insertion, and a deletion in PEN. Since ADJ takes only $O(m)$ storage, this yields

LEMMA 3.2. Let H be a set of n planes in E^3, with $A(H)$ simple, and let $S(E)$ be a skeleton of m edges in $A(H)$. $S(E)$ can be constructed in $O(m + S_I(n) + S_E(n) + S_P(n))$ storage and $O(T_I(n) + m(\log m + T_E(n) + T_P(n)))$ time.

By Proposition 3.1, $S_P(n) = O(n)$ and $T_P(n) = O(\sqrt{n} \log n)$ is achievable. In all applications treated in this paper, we also have $S_E(n)$ and $T_E(n)$ constant, and $S_I(n) = O(n)$. In this case, the complexities are $O(m + n)$ storage and $O(T_I(n) + m\sqrt{n} \log n)$ time.

4. Coping with Degenerate Cases. We now come back to the general case which deals with arbitrary, so possibly non-simple arrangements. The degeneracy that hurts Algorithm 2 the most occurs if more than three planes intersect in a common point. We remove all degeneracies artificially by simulation of a controlled perturbation:

> the parameters of each plane are changed into polynomials in ε, with ε a positive real number that remains without exact specification.

We simulate the construction of the perturbed arrangement resting on the choice of ε which must be sufficiently small but can be arbitrarily small.

Let $H = \{h_0, \ldots h_{n-1}\}$ be a set of non-vertical planes in E^3, with $A(H)$ not necessarily simple. We derive a simple arrangement $A(H(\varepsilon))$ which retains all important combinatorial properties of $A(H)$:

> For $h_i: x_3 = a_i x_1 + b_i x_2 + c_i$ in H, define $h_i(\varepsilon)$: $x_3 = a_i' x_1 + b_i' x_2 + c_i'$, with $a_i' = a_i + \varepsilon^{2^{3i+2}}$, $b_i' = b_i + \varepsilon^{2^{3i+1}}$, and $c_i' = c_i + \varepsilon^{2^{3i}}$, for $0 \le i \le n-1$. Then $H(\varepsilon) = \{h_i(\varepsilon) | 0 \le i \le n-1\}$.

Notice that the amount of perturbation experienced decreases double-exponentially with increasing index. We show

LEMMA 4.1. *$A(H(\varepsilon))$ is simple if ε is small enough.*

PROOF. We first argue that any three planes intersect in exactly one point, then we show that no four planes have a point in common.

Three planes $h_i(\varepsilon)$, $h_j(\varepsilon)$, $h_k(\varepsilon)$ intersect in a common point if and only if their normal vectors are linearly independent, that is, if and only if

$$\det \begin{pmatrix} a_i' & b_i' & -1 \\ a_j' & b_j' & -1 \\ a_k' & b_k' & -1 \end{pmatrix} \neq 0,$$

which is trivially true if ε is sufficiently small.

Four planes with indices i, j, k, l intersect in a common point only if

$$(*) \qquad \det \begin{pmatrix} a_i' & b_i' & c_i' & -1 \\ a_j' & b_j' & c_j' & -1 \\ a_k' & b_k' & c_k' & -1 \\ a_l' & b_l' & c_l' & -1 \end{pmatrix} = 0$$

which is impossible for small enough ε. □

(a) (b)

Fig. 2. Perturbing a non-simple arrangement. (a) Degenerate case. (b) Perturbed arrangement.

To substantiate that $H(\varepsilon)$ is a useful substitute for H, we demonstrate that $A(H(\varepsilon))$ and $A(H)$ are combinatorially related in a strong sense, and that the simulation of $A(H(\varepsilon))$ is inexpensive. Some notation is introduced first.

A face f in $A(H)$ defines a partition of H into sets $H_f^+ = \{h | f \text{ in } h^+\}$, $H_f = \{h | f \text{ in } h\}$ and $H_f^- = \{h | f \text{ in } h^-\}$. Similarly, a face g in $A(H(\varepsilon))$ defines a partition of H into sets H_g^+, H_g, H_g^- such that h is in H_g^+, H_g, H_g^- if g is in $h(\varepsilon)^+$, $h(\varepsilon)$, $h(\varepsilon)^-$, respectively. Now g is in the *perturbation of* f if H_f is minimal such that

$$H_f^+ \subseteq H_g^+, \; H_f^- \subseteq H_g^-, \text{ and } H_g \subseteq H_f.$$

Intuitively, the perturbation of f consists of all faces in $A(H(\varepsilon))$ which collapse into f when ε approaches zero. We illustrate the concept for the two-dimensional arrangements of lines shown in Fig. 2. All bounded faces of the perturbed arrangement (Fig. 2(b)) are in the perturbation of the only vertex in the non-perturbed arrangement (Fig. 2(a)). The perturbations of all other faces have cardinality one. Tedious but otherwise straightforward arguments show

OBSERVATION 4.2. Let H be a finite set of planes in E^3, and let $\varepsilon > 0$ be sufficiently small.

(i) The perturbations of faces in $A(H)$ define a partition of the faces in $A(H(\varepsilon))$.
(ii) Faces f_1 and f_2 in $A(H)$ are incident if and only if there are faces g_1 and g_2 in the perturbations of f_1 and f_2, respectively, with g_1 and g_2 incident.

By Observation 4.2(i), the following definition makes sense: Let E' be some set of edges in $A(H(\varepsilon))$. An edge e of $A(H)$ is in the *concentration* E *of* E' if there is an edge of E' in the perturbation of e. By Observation 4.2(ii), $S(E)$ is a skeleton in $A(H)$ if $S(E')$ is one in $A(H(\varepsilon))$.

The simulation of $A(H(\varepsilon))$ in the construction of a skeleton $S(E)$ in $A(H)$ consists of three parts:

1. Define a skeleton $S(E')$ in $A(H(\varepsilon))$ with E the concentration of E'.
2. Simulate computations in $A(H(\varepsilon))$ although ε remains without exact specification.
3. Obtain $S(E)$ from $S(E')$.

Part 1 depends on the particular application and will not be discussed in this section. Part 3 is straightforward and can be performed by repeated eliminating or merging of vertices and edges. The only operations of Algorithm 2 affected by the perturbation of $A(H)$ are the penetration queries which now yield unique answers since $A(H(\varepsilon))$ is simple. The simulation can be performed on the basis of the possibility to choose ε arbitrarily small, but positive:

> The only primitive operation needed to answer penetration queries involves a plane (say $h_i(\varepsilon)$) and a vertex v (the intersection of three planes, say $h_j(\varepsilon)$, $h_k(\varepsilon)$, $h_l(\varepsilon)$), and decides whether v is above or below $h_i(\varepsilon)$; by perturbation, $h_i(\varepsilon)$ cannot contain v. To decide the case, however, it is enough to determine whether the determinant in $(*)$ is positive or negative. This determinant is a polynomial in ε and the decision can be based on the sign of the first non-zero coefficient if the terms are ordered in increasing exponents. (Details of the method will appear elsewhere.)

Such a decision can still be done in constant time although some care is necessary to keep the constant small. The increase in storage is negligible since a polynomial can be constructed from three indices.

We now recalculate the requirements for the construction of a skeleton (see Lemma 3.2) in an arrangement $A(H)$ of n planes. We assume that the skeleton is defined for any finite set of nonvertical hyperplanes, that is, $E = E(H)$, and that $E(H)$ is the concentration of $E(H(\varepsilon))$ (which is true for all applications in this paper). Recall that $S_I(n)$, $T_I(n)$, $S_E(n)$, $T_E(n)$, $S_P(n)$, and $T_P(n)$ denote the requirements in storage and time for finding an initial vertex, determining incident edges, and processing penetration queries or insertions and deletions of halfspaces.

THEOREM 4.3. Let H be a set of n planes in E^3, $S(E(H))$ a skeleton in $A(H)$, and $m(n)$ the maximum cardinality of $E(H)$, over all sets H of n planes. $S(E(H))$ can be constructed in $O(m(n) + S_I(n) + S_E(n) + S_P(n))$ storage and $O(T_I(n) + m(n)(\log m(n) + T_E(n) + T_P(n)))$ time.

Again, we remark that for all applications in this paper, the requirements are $O(m(n))$ storage and $O(m(n)\sqrt{n}\log n)$ time.

5. Order-k Power Diagrams. Let s be a circle with center $c = (c_1, c_2)$ and radius $r \geq 0$ in E^2. For $p = (p_1, p_2)$ a point in E^2, $d_s(p) = (p_1 - c_1)^2 + (p_2 - c_2)^2 - r^2$ is called the *power* or the *Laguerre distance of p from s*. If s is a point then $d_s(p)$ is the square of the Euclidean distance between s and p. If p is outside s then $d_s(p)$ is also the length squared of the segment on a tangent through p which connects p and the point of s where the tangent touches s.

For S a set of n circles in E^2 and $S' \subseteq S$, $\text{dom}(S') = \{x \in E^2 | d_s(x) < d_t(x), s \in S', t \in S - S'\}$ is termed the *domination of* S'. If we define $\text{dom}(s, t) = \{x \in E^2 | d_s(x) < d_t(x)\}$ then

$$\text{dom}(S') = \bigcap_{s \in S', t \in S - S'} \text{dom}(s, t)$$

and dom(S') is a convex polygon since dom(s, t) is a halfplane. The collection of all non-empty dominations for subsets of S with cardinality k ($1 \leq k \leq n-1$) define the *order-k power diagram k-PD(S) of S*.

To construct *k-PD(S)*, we use a geometric transform E which maps each circle in S into a plane in E^3 and relates *k-PD(S)* with some particular skeleton in the arrangements. More specifically, there is a skeleton such that *k-PD(S)* can be obtained by projecting it vertically onto the plane $x_3 = 0$:

for s a circle with center $c = (c_1, c_2)$ and radius r, E maps s into the plane $E(s)$: $x_3 = 2c_1x_1 + 2c_2x_2 + (r^2 - c_1^2 - c_2^2)$.

Intuitively, E identifies E^2 with the plane $x_3 = 0$ in E^3, and maps s into the plane $E(s)$ which intersects the paraboloid U: $x_3 = x_1^2 + x_2^2$ in the vertical projection of s onto U. For a point $p = (p_1, p_2)$ in E^2 let $p_U = (p_1, p_2, p_1^2 + p_2^2)$ and $p_{E(s)} = (p_1, p_2, 2c_1p_1 + 2c_2p_2 + (r^2 - c_1^2 - c_2))$ be the vertical projections of p onto U and $E(s)$, respectively. Then we have

OBSERVATION 5.1. Let s and p be a circle and a point in E^2. Then $d_s(p)$ equals the (vertical) distance between p_U and $p_{E(s)}$.

Define $H = E(S) = \{E(s) | s \in S\}$, and for any point p in E^3, let $a(p)$, $o(p)$, $b(p)$ denote the number of planes h in H with p in h^-, h, h^+, respectively. Obviously, $a(p) + o(p) + b(p) = n$, the cardinality of H. Without details we state

OBSERVATION 5.2. Let S be a set of n circles in E^2, $H = E(S)$, and k some integer with $1 \leq k \leq n-1$.

(i) The set E_k of edges in $A(H)$ with $a(p) \leq k-1$ and $b(p) \leq n-k-1$ define a skeleton $S(E_k)$ in $A(H)$.
(ii) The vertical projection of $S(E_k)$ onto $x_3 = 0$ yields *k-PD(S)*.

We refer to [A, ES] for details on the correspondence between $S(E_k)$ and *k-PD(S)*. To construct $S_k(E)$, we use Algorithm 2 outlined in Section 3 and remove degeneracies by the methods of Section 4. This allows us to assume that $o(p) = 2$, for any point p of any edge in $A(H)$, and that any fixed (and non-perturbed) line in E^3 intersects the planes of H in n distinct points. To complete the specification of Algorithm 2 we demonstrate

1. how the edges in $S(E_k)$ incident upon a vertex v in $S(E_k)$ can be determined in constant time, if one incident edge e and the three planes f, g, h in H that define v are given, and
2. how an initial vertex of $S(E_k)$ can be found in linear time.

Task 1 is trivial since f, g and h intersect in three lines giving rise to six edges incident upon v. Among the six edges, three are below one of f, g, h, and the other three have none of the planes above them. The two edges in the same class with e are in $S(E)$.

The remainder of this section elaborates on the second task and concludes with the main result. To compute an initial vertex of $S(E_k)$, we use

OBSERVATION 5.3. Let S, H, and $S(E_k)$ be as defined above and let p be a point in E^3. p is a vertex of $S(E_k)$ if and only if $o(p) = 3$ and $a(p) = k - 1$ or $k - 2$.

Intuitively, the algorithm computes a line, intersects it with all planes, and finds a point close to a desired vertex. This vertex is then determined by setting up a new line through the point.

Algorithm 3 (Initial vertex for $S(E_k)$).

Step 1: Let $h^* = E(s^*)$ such that s^* in S has the rightmost center $c = (c_1, c_2)$ (that is, c_1 is maximal).

Step 2: Let g be the plane $x_2 = 0$, define line $\ell = h^* \cap g$, and intersect ℓ with all planes in $H - \{h^*\}$. {*Comment*: ℓ is the steepest line in $\{h \cap g | h \in H\}$, therefore $a(p_i) = i - \ell$ and $o(p_i) = 2$, if $p_1, \ldots p_{n-1}$ is the sequence of intersections on ℓ, sorted by decreasing x_1-coordinates.}

Step 3: Determine p_k and plane h^0 with $p_k = \ell \cap h^0$. {*Comment*: p_k is contained in an edge of $S(E_k)$ supported by $h^* \cap h^0$.}

Step 4: Intersect $h^* \cap h^0$ with all other planes of H and find the intersection v closest to p_k. {*Comment*: v is a vertex of $S(E_k)$.}

$O(n)$ time suffices for all four steps of Algorithm 3. By Theorem 4.3, we conclude

THEOREM 5.4. Let S be a set of n circles in E^2, and define $m_k(n)$ as the maximum number of edges in the order-k power diagram of n circles. There is an algorithm which constructs k-$PD(S)$ in $O(m_k(n))$ storage and $O(m_k(n)\sqrt{n} \log n)$ time.

For the case $k = 1$ or $n - 1$, E_k is the set of edges of a convex polyhedron in E^3 which can be constructed in $O(n \log n)$ time [A, PH], thus outperforming our method. However, if $k > 1$, the outlined algorithm is the first one that avoids the expensive construction of the arrangement $A(H)$ ($H = E(S)$) [EOS], which takes $O(n^3)$ time and storage. Unfortunately, no good upper bounds on $m_k(n)$ are presently known except for

$$m_1(n) = n \quad \text{which is trivial,}$$

$$m_2(n) = 3n - 6 \text{ [ESt], and}$$

$$m_k(n) = O(nk^5) \text{ [CP].}$$

In the special case when S is a set of points in E^2, k-$PD(S)$ is known as the *order-k Voronoi diagram k-VOD(S) of S* [SH]. For this case $m_k(n) =$

$O(k(n-k))$ and an algorithm that constructs k-$VOD(S)$ in $O(k^2n)$ storage and $O(k^2n \log n)$ time is known [L]. Our algorithm is more space efficient and asymptotically faster unless $k = O(\sqrt{n})$.

6. Quartering Separated Point-Sets in E^3. For the sake of this section's definitions we introduce the concept of a directed plane h in E^3 given by a normal vector v and some real number α, that is, h consist of all points x with $\langle x, v\rangle = \alpha$. h^p: $\langle x, v\rangle > \alpha$ and h^n: $\langle x, v\rangle < \alpha$ are called the *positive* and *negative sides of* h, respectively. h is said to bisect a set P of n points in E^3 if

$$\max\{|P \cap h^p|, |P \cap h^n|\} \leq n/2.$$

Two sets P and Q in E^3 are said to be *separable* if $P \subseteq h^p$ and $Q \subseteq h^n$ for some directed plane h. [YDE] show

PROPOSITION 6.1. Let P and Q be separable sets of m and n points in E^3, respectively. There are directed planes g and h which both bisect P and Q such that $\max\{|P \cap g^p \cap h^p|, |P \cap g^p \cap h^n|, |P \cap g^n \cap h^p|, |P \cap g^n \cap h^n|\} \leq m/4$, and $\max\{|Q \cap g^p \cap h^p|, |Q \cap g^p \cap h^n|, |Q \cap g^n \cap h^p|, |Q \cap g^n \cap h^n|\} \leq n/4$.

A straightforward perturbation argument shows that g and h can be restricted to planes f (termed *spanned bisectors*) which satisfy

(i) $|P \cap f| \geq 1$ and $|Q \cap f| \geq 1$, and
(ii) not all points in $(P \cup Q) \cap f$ are collinear.

For convenience, we assume that no three points of $P \cup Q$ are collinear; in this case, (ii) is equivalent to requiring that f contains at least three points. Further justification of this assumption is provided by the methods of Section 4 which simulate simplicity if degeneracies occur.

We now develop a correspondence between all spanned bisectors of P and Q and the vertices of a particular skeleton in an arrangement $A(G \cup H)$ dual to $P \cup Q$. Sets G and H of planes in E^3 can be obtained by means of the following transform D:

> For $p = (p_1, p_2, p_3)$ a point in E^3, $D(p)$ is a plane defined by $x_3 = 2p_1x_1 + 2p_2x_2 - p_3$. Conversely, we write $D(g) = D^{-1}(g)$, for a non-vertical plane g. Then $G = D(P)$ $(= \{g | g = D(p), p \in P\})$ and $H = D(Q)$.

$A(H \cup G)$ is dual to $P \cup Q$ because of

OBSERVATION 6.2. Let p be a point and h a non-vertical plane in E^3. Then p is in h^+, h, h^- if and only if point $D(h)$ is in $D(p)^+$, $D(p)$, $D(p)^-$, respectively.

For any point q in E^3, we define $a_G(q)$, $o_G(q)$, $b_G(q)$ $(a_H(q), o_H(q), b_H(q))$ as the number of planes g in G (H) with q in g^-, g, g^+, respectively. We now have

OBSERVATION 6.3. Let h be a non-vertical plane in E^3 and write $q = D(h)$. h is a spanned bisector of P and Q if

(i) $\max\{a_G(q), b_G(q)\} \leq |G/2|$ and $o_G(q) \geq 1$,
(ii) $\max\{a_H(q), b_H(q)\} \leq |H/2|$ and $o_H(q) \geq 1$, and
(iii) $o_G(q) + o_H(q) \geq 3$.

We now show that conditions (i) and (ii) define a skeleton in $A(G \cup H)$ if the sets of planes, respectively their dual sets of points, satisfy the following conditions:

1. $|P|$ and $|Q|$ are odd, and
2. all points in P (Q) have negative (positive) x_1-coordinates.

Conditions 1 and 2 are no loss of generality by the following arguments: if $|P|$ ($|Q|$) is even then add an arbitrary point to P (Q) which does not violate condition 2 and the separability of P and Q; a plane bisects the new set only if it bisects the old one. To achieve 2, apply a suitable rotation and translation to $P \cup Q$.

LEMMA 6.4. Let P and Q be separable point-sets in E^3 which satisfy conditions 1 and 2 above, define $G = D(P)$ and $H = D(Q)$, and let $E_{G,H}$ contain all edges in $A(G \cup H)$ whose points q satisfy (i) and (ii) from Observation 6.3. Then $S(E_{G,H})$ is a skeleton.

PROOF. We first show that any plane f normal to the x_2-axis intersects $S(E_{G,H})$ in a unique point. Define

$$G_f = \{f \cap g | g \in G\} \text{ and } H_f = \{f \cap g | g \in H\}.$$

$A(G_f)$ and $A(H_f)$ are two arrangements of lines in f. All lines in G_f (H_f) have negative (positive) slope if we use the x_1-axis (x_3-axis) to define the horizontal (vertical) direction in f. Define the *median level in* $A(G_f)$ ($A(H_f)$) as the set of points q in f which satisfy condition (i) (condition (ii)) of Observation 6.3. Since $|G_f|$ and $|H_f|$ are odd, both median levels are continuous functions from x_1 to x_3, and they intersect exactly once since they consist of a finite number of linear pieces, and one monotonely decreases while the other monotonely increases. To trace all points of $S(E_{G,H})$, move plane f continuously through the three-dimensional arrangement. Since the lines in plane f move continuously at the same time, the intersection of the two median levels is a continuously moving point which implies the connectivity of $S(E_{G,H})$. □

To construct $S(E_{G,H})$ using Algorithm 2, we need to specify how an initial vertex can be determined, and, given a vertex v with incident edge and defining planes, how all edges of $S(E_{G,H})$ incident upon v can be found. The latter task is trivial since v is the intersection of exactly three planes (see Section 4). An initial vertex can be found by intersecting the median levels in $A(G_f)$ and $A(H_f)$, for f: $x_2 = 0$, say (see the proof of Lemma 6.4). The determined point defines a vertex or an edge of $S(E_{G,H})$; in the latter case an endpoint is trivially computed in

linear time. By [M], there is an algorithm that intersects the two median levels in $O(|G| + |H|)$ time; a suboptimal method can be found in [CSY]. If we write $t(n)$ for the maximum number of edges in $S(E_{G,H})$, for $n = |G| + |H|$, then Theorem 4.3 implies

THEOREM 6.5. Let P and Q be separable sets of a total of n points in E^3. There is an algorithm that constructs $S(E_{G,H})$, with $G = D(P)$ and $H = D(Q)$, in $O(t(n)\sqrt{n}\log n)$ time and $O(t(n))$ storage.

From $S(E_{G,H})$, a pair of planes satisfying Proposition 6.1 can be determined by testing all pairs of spanned bisectors which appear as vertices of $S(E_{G,H})$. We keep one vertex fixed and test all other vertices in sequence. If we also keep track of the numbers of points in the four wedges defined by the corresponding planes, constant time suffices for a test. The $O(t(n))$ bound for the needed storage can be improved to $O(n)$ if, at any point in time, only two connected pieces of $S(E_{G,H})$ with respective n vertices are kept in storage. For each first piece $t(n)/n$ second pieces are constructed and pairs of vertices are tested. This strategy builds up $S(E_{G,H})$ $t(n)/n + 1$ times and needs $O(t^2(n))$ time for the various pairs of vertices. We conclude

THEOREM 6.6. Let P and Q be separable sets of a total of n points in E^3. There is an algorithm which finds two planes g and h with at most $|P|/4$ ($|Q|/4$) points of P (Q) in each open wedge in $O(t^2(n))$ time and $O(n)$ storage.

The result of this section suffer from our insufficient knowledge about $t(n)$. To aid future combinatorial investigations of $t(n)$, we offer a more intuitive definition. Let P and Q be two separable sets of points in E^3 such that $|P|$ and $|Q|$ are odd. If $n = |P| + |Q|$, then $t(n)$ equals the number of spanned bisectors of P and Q. Clearly $t(n) = O(n^3)$, and also $t(n) = \Omega(n \log n)$ follows from combinatorial results on two-dimensional point-sets reported in [ELSS, EW]. We conjecture that the lower bound is closer to the truth than the trivial upper bound.

7. Discussion. The main contribution of this paper is the introduction of the rather general concept of skeletons in arrangements of planes in E^3 and the presentation of a method that constructs skeletons. The crucial components of the method are, first, a local algorithm for tracing an eulerian tour in a particular kind of directed graphs, and second, a data structure for finding a plane that is hit first by a linearly moving point. The serious problems that occur with degenerate cases are resolved by a controlled simulation of non-degeneracy; the author believes that this simulation method is widely applicable in the field of geometric computation.

The presented method for constructing skeletons in three-dimensional arrangements leads to new and more efficient algorithms for several problems in computational geometry: Section 5 demonstrates the construction of order-k power diagrams for sets of circles in E^2 (if all circles are congruent then the

diagram coincides with the order-k Voronoi diagram for the centers). The construction of two planes that cut two separable point-sets in E^3 into respective four balanced parts is described in Section 6. Along similar lines, skeletons can be used to solve e.g. the following related problems:

(1) Compute the *k-degree power diagram* for a set of n circles in E^2, with $1 \leq k \leq n-1$, defined as follows: each circle s in the sets gets associated the region of points p such that $d_s(p)$ is the k-th smallest, for all circles.
(2) Determine the *region of centerpoints* of a set P of n points in E^3. (A point p is *centerpoint of* P if any closed halfspace bounded by a plane through p contains at least $\lceil n/4 \rceil$ points of P.)
(3) For three sets P, Q, R of points in E^3 compute a *ham-sandwich plane* h, that is, h bisects P, Q, and R.

By methods similar to those in Section 6, problem (3) can be solved in $O(t^2(n))$ time and $O(n)$ storage if P, Q, and R contain a total of n points and any two are separable from each other. Together with Theorem 6.6, this yields an $O(t^2(n))$ time and linear space algorithm for the type of space cutting trees described in [YDE].

Acknowledgements. The author thanks Ernst Muecke for implementing parts of the algorithms described in this paper, for simplifying parts of them, and for many interesting discussions on the material.

References

[AHU] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The design and analysis of computer algorithms*. Addison Wesley, 1974.

[AW] G. L. Alexanderson and J. E. Wetzel, Simple partitions of space. *Math. Mag.* 51 (1978), 220–225.

[A] F. Aurenhammer, Power diagrams: properties, algorithms, and applications. Rep. F 120, Inst. Inform. Process., Techn. Univ. Graz, Austria, 1983.

[CP] B. M. Chazelle and F. P. Preparata, Halfspatial range search: an algorithmic application of k-sets. To appear in *J. Discrete Comput. Geom.*

[CSY] R. Cole, M. Sharir and C. K. Yap, On k-hulls and related problems. *Proc. 16th Ann. ACM Symp. Theor. Comput. Sci.* (1984), 154–166.

[DE] D. P. Dobkin and H. Edelsbrunner, Space searching for intersecting objects. *Proc. 25th Ann. IEEE Symp. Found. Comput. Sci.* (1984), 387–392.

[EH] H. Edelsbrunner and F. Huber, Dissecting sets of points in two and three dimensions. Rep. F 138, Inst. Inform. Process., Techn. Univ. Graz, Austria, 1984.

[EM] H. Edelsbrunner and H. A. Maurer, Finding extreme points in three dimensions and solving the post-office problem in the plane. *Inform. Process. Lett.* 21 (1985), 39–47.

[EOS] H. Edelsbrunner, J. O'Rourke and R. Seidel, Constructing arrangements of lines and hyperplanes with applications. *Proc. 24th Ann. IEEE Symp. Found. Comput. Sci.* (1983), 83–91.

[ES] H. Edelsbrunner and R. Seidel, Voronoi diagrams and arrangements. *Proc. Sympos. Comput. Geom.*, Baltimore (1985), 251–262.

[ESt] H. Edelsbrunner and G. Stoeckl, The number of extreme pairs of finite point-sets in Euclidean spaces. Rep. F 142, Inst. Inform. Process., Techn. Univ. Graz, Austria, 1984.

[EW] H. Edelsbrunner and E. Welzl, On the number of line separations of a finite set in the plane. *J. Combin. Theory Ser. A* 38 (1985), 15–29.

[ELSS] P. Erdoes, L. Lovasz, A. Simmons and E. G. Straus, Dissection graphs of planar point-sets. In: *A survey of combinatorial theory*, J. N. Srivastava et al. (eds.), North Holland, 1973, 139–149.

[GK] I. G. Gowda and D. G. Kirkpatrick, Exploiting linear merging and extra storage in the maintenance of fully dynamic geometric data structures. *Proc. 18th Ann. Allerton Conf. Commun., Contr., Comput.* (1980), 1–10.

[G] B. Gruenbaum, Arrangements and hyperplanes. *Congr. Number.* III, *Louis. Conf. Combin., Graph Th., Comput.* (1971), 41–106.

[IIM] H. Imai, M. Iri and K. Murota, Voronoi diagram in the Laguerre metric and its applications. *SIAM J. Comput.* 14 (1985), 93–105.

[K] D. E. Knuth, *Searching and sorting. The art of computer programming* III. Addison-Wesley, 1973.

[L] D. T. Lee, On k-nearest neighbour Voronoi diagram in the plane. *IEEE Trans. Comput.* C-31 (1982), 478–487.

[MO] H. A. Maurer and Th. Ottmann, Dynamic solutions of decomposable searching problems. In: *Discrete structures and algorithms*, U. Pape, (ed.), Carl Hanser (1979), 17–24.

[M] N. Megiddo, Partitioning with two lines in the plane. To appear in *J. Algorithms.*

[PH] F. P. Preparata and S. J. Hong, Convex hulls of finite sets of points in two and three dimensions. *Comm. ACM* (1977), 87–93.

[SH] M. I. Shamos and D. Hoey, Closest-point problems. *Proc. 16th Ann. IEEE Symp. Found. Comput. Sci.* (1975), 151–162.

[vLW] J. van Leeuwen and D. Wood, Dynamization of decomposable searching problems. *Inform. Process. Lett.* 10 (1980), 51–56.

[YDE] F. F. Yao, D. P. Dobkin and H. Edelsbrunner, Manuscript, 1985.